VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

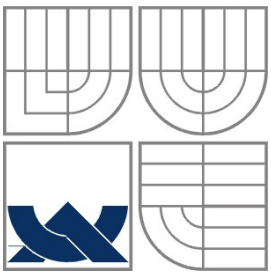
OVLÁDÁNÍ APLIKACE GESTY RUKOU A PRSTŮ UŽIVATELE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

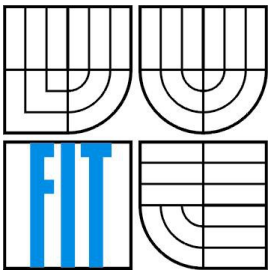
AUTOR PRÁCE
AUTHOR

MARTIN BŘENEK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OVLÁDÁNÍ APLIKACE GESTY RUKOU A PRSTŮ UŽIVATELE

APPLICATION CONTROL USING USER HAND AND FINGERTIPS GESTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BŘENEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2014

Abstrakt

Tato bakalářská práce se zabývá tvorbou prvku uživatelského rozhraní, který slouží k ovládání aplikace za pomoci gesta tvořeného rukou a prsty uživatele. V textu jsou rozebrány základy počítačového vidění a principy segmentace obrazu. Dále se práce zaměřuje na analýzu zadání a návrh, popisuje implementační detaily, včetně použitých nástrojů a zahrnuje výsledky testů koncového řešení.

Abstract

This bachelor's thesis deals with creating the user interface element, which is used to control application by user hand and fingertips gesture. The text discuss the basics of computer vision and image segmentation principles. The thesis also focuses on analysis of task and solution design, describes implementation details, including used tools and includes test results of final solution.

Klíčová slova

Počítačové vidění, uživatelské rozhraní, HCI, segmentace obrazu, Kinect, hloubkový obraz, OpenCV, ROS.

Keywords

Computer vision, user interface, HCI, image segmentation, Kinect, depth image, OpenCV, ROS.

Citace

Martin Břenek: Ovládání aplikace gesty rukou a prstů uživatele, bakalářská práce, Brno, FIT VUT v Brně, 2014

Ovládání aplikace gesty rukou a prstů uživatele

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Břenek
20. května 2014

Poděkování

Chtěl bych poděkovat panu Ing. Vítězslavu Beranovi, Ph.D. za odbornou pomoc a cenné rady, které mi pomáhaly v průběhu realizace celé práce. Také bych chtěl poděkovat Michalu Chludovi za pomoc při testování.

Obsah

1 Úvod

Tento dokument je technickou zprávou k bakalářské práci s názvem „Ovládání aplikace gesty rukou a prstů uživatele.“ Čtenář může v tomto dokumentu najít základní teoretické poznatky o řešeném problému, následný návrh řešení, jeho částečnou implementaci a informace, o již provedených testech.

Účelem této práce je vytvořit prvek uživatelského rozhraní pro ovládání vybrané aplikace. V tomto případě nejsou slovem ovládání myšlena klasická periferní zařízení, jako je například myš či klávesnice u PC, nýbrž gesta, která jsou tvořena pomocí rukou, speciálně dlaní a konečky prstů uživatele. Výsledný prvek by měl být nástrojem, který usnadňuje práci s PC takovým způsobem, že omezuje používání klasických ovládacích prvků zmíněných výše a soustřeďuje se na bezdotykovou komunikaci, známou pod zkratkou HCI (human-computer interaction).

Dnešní doba je symbolem pokroku v podstatě ve všech vědních disciplínách. Lidé jsou přizpůsobiví a hledají si cestu k moderním součástem našeho světa. Příkladně HCI je obor, který má v budoucnosti co nabídnout, a to nejenom využitím bezdotykové komunikace v průmyslu a úzce zaměřených oblastech, ale i obyčejným lidem v zefektivnění a usnadnění jejich práce. Hodně lidí si bezdotykovou komunikaci vyzkoušelo na vlastní kůži. Jsou to ale převážně děti, mladí lidé nebo nadšenci do herního světa, kde ovládání her pomocí snímání pohybu uživatele, je už nějakou dobu úspěšně používáno.

Využití tohoto prvku je široké. Není to jen herní průmysl a jemu podobná odvětví, ale jsou to také lidé s postižením, které omezuje používat klasické ovládací prvky. Další příkladným užitím je možnost aplikovat řešení do zařízení na veřejných místech. Rychlejší a hygieničtější komunikace by se zajisté těšila vetší oblibě, než ta dosavadní.

Další rozbor řešené problematiky, základních znalostí, o kterých by si měl čtenář udělat povědomí k lepšímu pochopení navrženého řešení, jsou konkrétněji vysvětleny v teoretické části. Ze získaných poznatků byl vytvořen návrh řešení. Jeho konkrétní části a body jsou k nalezení v kapitole Návrh řešení. Vytvořený koncept byl uveden do praxe. Jakých nástrojů, případně zařízení, se při tom použilo se můžete dočíst v sekci Realizace a Testování. Prvek prošel různými testy ať již v průběhu implementace nebo jako hotový. Popis způsobu metodiky testování a získaných výsledků obsahuje také kapitola Realizace a Testování. Závěr obsahuje informace o celkovém zhodnocení výsledku práce. Autor se také zamýšlí nad možnostmi vylepšení stávajícího řešení, případně úpravami, tak, aby prvek vyhovoval použití.

2 Teorie

K vývoji nových prvků uživatelského rozhraní zaměřených na komunikaci mezi počítačem a člověkem je nutné se seznámit s obecnými pojmy, používanými trendy a již vytvořenými produkty z této oblasti. K řešení této práce byly využity poznatky získané studiem zásad tvorby uživatelských rozhraní. Následující část popisuje vědní disciplíny jako je počítačové vidění včetně rozboru obrazových segmentačních technik a interakce člověk-počítač (HCI)

2.1 Počítačové vidění

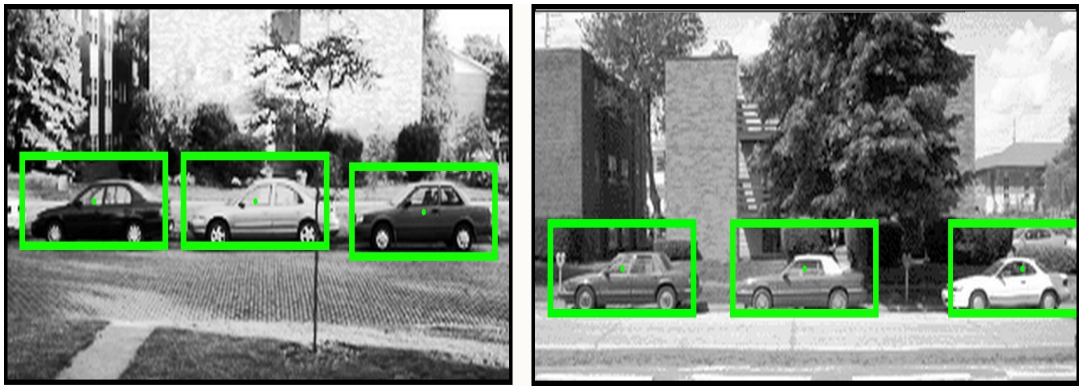
Tato veskrze nová oblast se zabývá vývojem software a vytvářením produktů, které dokážou získávat a interpretovat ze snímaného obrazu vizuální informaci [1] . Obor je to velice obsáhlý a neustále se rozšiřuje. Můžeme jmenovat spousty odvětví, kde se počítačové vidění používá. Jsou to například:

- **Ovládání procesů** – v současné době se téměř všude ve výrobních procesech využívá automatizované výroby (průmyslových robotů apod.)
- **Organizace informací** – indexace souborů obrázku, jejich třídění
- **Detekce jevů v reálném světě** – při experimentech v laboratořích, sledování událostí
- **Modelování objektů a prostředí** – v analýze obrazů ze zařízení používaných v medicíně
- **Interakce** – využití v oblasti HCI při zpracování vstupních informací

Každá z těchto jmenovaných oblastí definuje a využívá množství metod, které slouží ke zpracování nebo získávání dat úlohami danými užitím v konkrétní oblasti. Ve vztahu k zadání problému jsou zmíněny dvě z těchto úloh, tzv. rozpoznávání a analýzu pohybu.

Rozpoznávání je úloha, která se snaží zjistit, jestli určená data obsahují určitou vlastnost, činnost nebo objekt. Se zaměřením na úzkou skupinu vlastností, činností nebo objektů, je tato úloha řešitelná celkem snadno a s vysokým úspěchem. Problém ovšem nastává při aplikaci na obecný případ, kdy se mohou naleznout libovolné objekty v libovolných situacích. V současnosti se odrážíme od schopnosti vytvořit metody řešící jednoduché geometrické objekty, lidské tváře, vozidla, projev psaný rukou apod. v příznivých podmínkách daných orientací objektu, vhodným nasvícením nebo vhodnou barvou pozadí. Díky různému užití metod se poznávací úloha pro přehlednost dělí do několika variant:

- **Poznání objektů** – Metoda má k dispozici specifikační data o objektech, případně jejich třídách. Zjistí, zda daný objekt patří do určité třídy (například, že daný objekt je člověk, dům apod.). Řadíme sem klasifikaci objektů.
- **Identifikace** – Rozpoznání určité části objektu, jako je tvář, otisk prstu konkrétního člověka.
- **Detekce** – Prohledávání dat k zjištění výskytu konkrétní podmínky (viz. obr. 2.1). Patří sem detektory bankovek, mýtného na dálnicích a medicínských obrazů. Řadí se sem i detekce gest vytvořených uživatelem.



Obrázek 2.1: Detekce objektu ve scéně. Převzato z [12]

Druhou skupinu úloh tvoří metody využívané k analýze pohybu. V sekvenci po sobě jdoucích obrazů se tyto metody snaží monitorovat pohyb a odhadnout rychlost změny hodnot v jednotlivých bodech v obraze (případně regionech) :

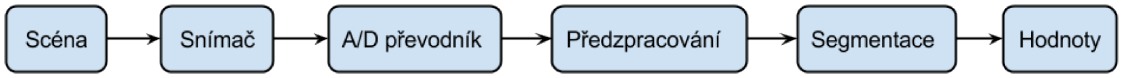
- **Egomotion** – Je metoda zabývající se odhadem pohybu objektu ve 3D scéně z pohledu objektu samotného. Pokud by byl objektem fotoaparát, určoval by se pohyb v prostoru pomocí sekvence snímků pořízených tímto fotoaparátem.
- **Tracking** – Metoda, která v sekvenci obrazových dat sleduje pohyb menšího souboru pozorovaných bodů. Ty mohou určovat sledovaný objekt.

Metody řešící výše zmíněné úlohy jsou sdružovány v tzv. Systémech počítačového vidění. Podle typu užití lze nalézt systémy pouze jako samostatné aplikace nebo i jako součást většího celku, který v sobě sdružuje i další podsystémy. Vnitřní implementace systému také reflektuje, zda je známa funkce systému předem, nebo je to systém s učením. Pohledem na celou škálu systému se dají odvodit typické funkce těchto systémů.

- **Získávání obrazu** – Snímání obrazové informace pomocí různých typů senzorů.
- **Předzpracování** – Surová data nasbíraná senzory ve většině případů neodpovídají požadavkům, aby na ně mohly být aplikovány metody počítačového vidění. Je nutné provést několik úkonů a data upravit.
- **Vybírání rysů** – Podle toho, co obrazová data zachytávají, obsahují tzv. rysy obrazu, které mají různou úroveň komplexnosti. Můžou to být jednoduché matematické tvary jako úsečky, čáry, křivky, hrany nebo lokalizované body jako rohy, kraje, kouty či jednotlivé body a poté komplexnější rysy založené na texturách, tvaru a pohybu.
- **Segmentace** – Jde o selekci oblastí obrázku, které mají určitý vztah s konkrétními objekty v obraze. Jsou důležité pro další zpracování obrázku.
- **Vysokoúrovňové zpracování** – V této části je na vstupu malé množství dat (seznamy bodů, části obrázku) a předpokládá se výskyt určitého objektu. S objektem se poté dále pracuje a testuje se, zda splňuje určitá pravidla (např. Pro klasifikaci zjištěných objektů do kategorií).

2.2 Zpracování obrazu

Zpracování obrazové informace se v praxi dělí na několik částí (viz. Obr. 2.2). Tyto části jsou realizovány pomocí různých zařízení. Pro snímání a převod se nejčastěji využívá hardwarového zařízení jako je kamera, pro následné předzpracování a segmentaci zase specializovaný software [2].



Digitalizace a snímání obrazu

Digitalizace obrazu je použití matematických funkcí k převodu obrazu reálného světa do interpretace srozumitelné pro počítač. Obraz je modelován pomocí obrazové funkce. Klasický statický 2D obraz je popsán obrazovou funkcí dvou souřadnic v rovině $f(x,y)$. Hodnoty dané funkce reprezentují předem danou veličinu. Příkladně pro barevný obraz je to intenzita každé z RGB složek, u termovize zase teplota. V digitálním světě je obraz reprezentovaný obrazovou maticí. Tato matice není nic jiného, než soubor hodnot reprezentovaný obrazovou funkcí. Jednotlivé prvky matice jsou nazývány pixely, což je nejmenší popsatelná jednotka obrazu.

Obrazová informace se získává pomocí různých typů a počtů senzorů. Kromě klasických senzorů, jako je rgb kamera a zařízení citlivých na světelný paprsek, existují také senzory vzdálenosti, radary, nadzvukové kamery nebo i tomograf. Každý typ senzoru má povětšinou rozdílnou interpretaci dat – výsledná data mohou být 2D obraz, 3D obraz nebo i sekvence 2D a 3D obrazů.S tím souvisí i jiné fyzikální veličiny, které tyto data reprezentují. Příkladně to může být hloubka, kterou dokáže měřit zařízení Kinect (viz kap. 2.5), absorbance nebo reflektance zvukových či elektromagnetických vln.

Předzpracování

Digitálně zpracovaná data snímačem a převodníkem je třeba dále analyzovat k získání potřebné informace o obsahu.

Získaný obraz však v tomto stádiu obsahuje deformace a vady kvůli technické nedokonalosti snímače. Minimalizaci těchto vad řeší tzv. Předzpracování. Při této technice se využívají hlavně poznatky z rekonstrukce obrazu pro odstranění nežádoucích jevů, jako je šum, zkreslení, špatný kontrast, prostorová deformace apod. Různými technikami od jednoduchých matematických filtrů až po složitější metody pracující separovaně s jednotlivými místy v obraze. Jeho aplikací je možné upravit obraz takovým způsobem, aby mohl být efektivně využíván vyššími úrovněmi zpracování [3].

Základní metody předzpracování obrazu můžeme rozdělit do těchto skupin.

- **Geometrické transformace**
- **Jasové transformace**
- **Úpravy filtrováním a ostřením**

Operace, které se provádí při předzpracování mají bohužel i vedlejší účinky. Například při odstranění šumu se obraz rozmaže a díky tomu mohou zaniknout jemné čáry a křivky, které jsou využívány hranovými detektory při následné segmentaci. Kromě výběru správných technik předzpracování je důležité zvolit míru s jakou vybranou techniku aplikujeme.

Mediánový filtr

Je to nelineární typ filtru, který je vhodný k odstranění náhodného šumu. Funguje na principu výběru mediánu. Pro každý pixel v obraze vybere jeho okolí a z hodnot okolních pixelů vybere střední hodnotu, medián, který se použije jako nová hodnota zkoumaného pixelu [4].

Segmentace

Na předzpracovaný obraz je aplikována segmentace [6] . Segmentace je jádrem a jedním z hlavních úkonů automatického zpracování obrazu. Je to proces, kde se dělí obraz do oblastí se společnými vlastnostmi, splňujících nějaké tvrzení. Tyto vzájemně se nepřekrývající oblastí buďto věrně kopírují objekty vstupního obrazu, kdy jde o kompletní segmentaci, a nebo se vytvořené segmenty se svou předlohou přesně neshodují, pak mluvíme o částečné segmentaci.

Ovšem kompletní segmentace, i s ohledem na pokročilé technologie a informace o obsahu analyzovaného obrazu, není realizovatelná. Tím pádem se zabýváme pouze částečnou segmentací. V tomto případě hledáme části, které jsou si do jisté míry podobné. Částečná segmentace se tedy využívá jako základ při analýze obrazových dat.

Informaci o dělení obrazu do segmentů, podobných částí, využívají vyšší algoritmy pro zpracování obrazu. Segmentačních technik (algoritmů) je veliké množství. Je podstatné zvolit a vybrat tu správnou, podle typu vstupních dat a záměru, ke kterému ji chceme využívat. Je proto dobré shrnout podle určitého způsobu klasifikace několik obecných technik s příklady určitých, v praxi používaných algoritmů.

- **Metody založené na detekci hran (edge-based methods)** – U těchto metod je snaha detekovat důležité hrany v obraze. Používají se tzv. Hranové detektory, jejichž algoritmus na základě rozdílu hodnot sousedních pixelů produkuje množinu hran v obraze. Patří sem například metoda aktivních kontur.
- **Metody založené na detekci oblastí (region-based methods)** – Tyto metody jsou velice podobné metodám založených na detekci hran. V ideálním případě by hrany detekované předchozím typem segmentace měly ohraničovat plochy (regiony) nalezené tímto způsobem. Jednou ze metod je tzv. šíření oblastí (region growing), do které spadá algoritmus Flood fill, dobře využitelný jako součást detektoru ruky v případě této práce.
- **Statistické metody** – analýza obrazových dat je statistická. Nejčastěji se analyzují hodnoty pixelů a zanedbává se se strukturní informace. Jako příklad lze uvést prosté, či adaptivní prahování.

A další pokročilé segmentační metody, jako jsou hybridní metody využívající matematickou morfologii nebo znalostní metody řídící se souborem předloh.
Následující blíže popsané segmentační metody byly vybrány na základě využití v projektu.

Prahování (Thresholding)

Jednoduchá metoda segmentace vyhodnocující hodnotu jasu každého pixelu. Pokud budeme uvažovat hloubkový obrázek v úrovni šedi (viz. obr. 2.3), tak rozložení této úrovně v obrazu nám může určit správný práh. Práh je hodnota, která se dá v ideálním stavu vyčíst z histogramu obrázku a oddělit takto dvě maxima, které histogram nabývá. Obecně je prahování funkce g, která transformuje vstupní obraz na výstupní černobílý (binární) obraz podle:

$$g(x,y)=\begin{cases}1 \; pokud \; f(x,y)\geq T\\ 1 \; pokud \; f(x,y)<T\end{cases}$$

kde T je hodnota prahu.

Pokud je hodnota jasu pixelu rovna, nebo vetší dané hodnotě prahu, bude pixel nabývat hodnoty 1, a pokud bude menší než práh, bude nabývat hodnoty 0. Tímto vznikne bitový obraz (viz. obr. 2.3), který je, co do velikosti reprezentujících dat, velice kompaktní a dá se použít jako vstup pro další metody segmentace.

V praxi ovšem ve většině případů není histogram ideální a neobsahuje ostré vrcholy a může mít několik rozličných vrcholů. V tomto případě je nutné použít jinou metodu, která poskytuje kvalitnější výsledky.

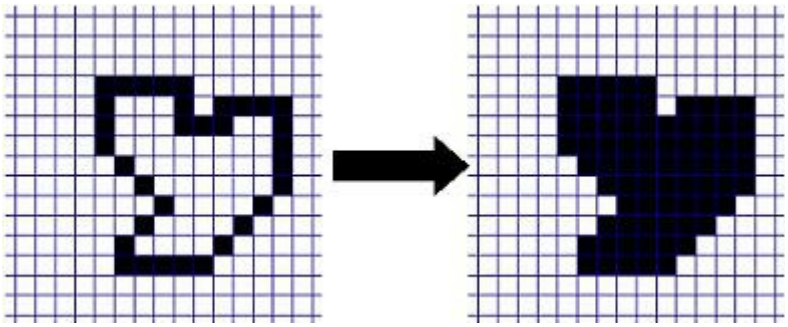
Obrázek 2.3: Obrázek ve stupních šedi a vyprahovaný obrázek.

Převzato z [14]

Vyplňovací algoritmus (Flood fill algorithm)

Tento algoritmus[5] , patřící do skupiny šíření oblastí (region growing), který je znám také pod jménem Seed fill, funguje na principu zaplavování oblastí s podobnými vlastnostmi ve vícerozměrném poli. Používá se k vyznačování a spojování oblastí pixelů se stejnou barvou. V případě standardní modifikace jsou k jeho funkčnosti potřebné tři parametry: startovací bod,

cílová barva a barva, za kterou se cílová barva vymění. Algoritmus započne svou funkci ve startovacím bodu (pixel v obraze) a postupně své sousední body (ve čtyř nebo osmi-okolí), které mají stejnou cílovou barvu, přebarvuje na barvu určenou v parametru funkce (viz. obr. 2.4). V praxi se používají různé modifikace, kdy například cílová barva není pouze jedna hodnota, ale určitý rozsah hodnot. Po této úpravě a výběrem vhodného rozsahu je algoritmus dobře použitelný k získání objektů a jejich částí na reálných datech, například obrázcích pořízených kamerou, či fotoaparátem.



Obrázek 2.4: Algoritmus Flood fill. Převzato z [13]

2.3 Klasifikace

Je procesem, který rozpoznává objekty v obraze [7] . Objekty jsou řazeny do předem známých tříd, což je soubor podobných si objektů, na základě rozhodovacího pravidla. Metody klasifikace se dělí do dvou základních skupin podle způsobu popisu objektů. Jsou to klasifikátory příznakové a strukturální. Příznakové fungují na principu využití příznaků, což můžeme chápat jako vektor číselných charakteristik objektu. Naproti tomu strukturální pracují s kvalitativním popisem rozpoznávaného objektu, kde jsou objekty popsány primitivy a za pomocí definované abecedy jazyka a gramatiky funguje rozpoznávání na principu rozboru slova a kontroly správnosti syntaxe.

Rozhodovací stromy

Jsou velice jednoduchou, rychlou a přehlednou technikou. Dobře se interpretují a díky tomu se z nich snadno vyhodnocují získané výsledky. Je to algoritmus, který se vytvoří z trénovací množiny (jde o učení s učitelem). Ta obsahuje objekty, jejich popis pomocí atributů a jejich rozřazení do tříd.

AdaBoost

Metoda, jejíž název vznikl z anglického „adaptive boosting“, je výsledkem vhodného spojení více slabších klasifikátorů, které jsou spolehlivé pouze na určité množině dat a na jejich vstupu využívají pouze jeden příznak (práh, histogram,...) v jeden silný klasifikátor. Je to nelineární klasifikátor a není příliš odolný vůči šumu.

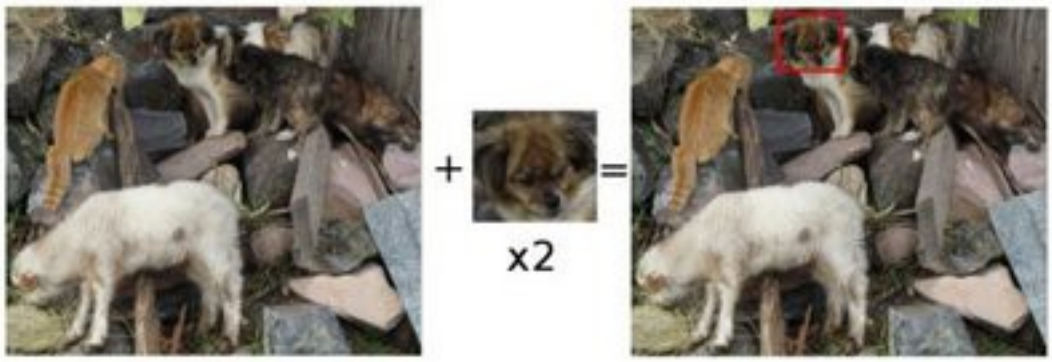
Support vector machines (SVM)

Je typem klasifikátoru, který rozděluje objekty do dvou tříd. Hledá se nadrovina, jež optimálně rozděluje data v prostoru příznaků. Cílem je vybrat takovou nadrovinu, aby hraniční body, jimiž prochází pomocné vektory (support vectors) vzdálené od nadroviny, byly co nejdále, resp. hodnota minima vzdáleností bodů byla co největší.

Template matching

Jsou metody klasifikace objektů v obraze na základě porovnávání. Základem pro tento typ klasifikace je maska, kterou může být například výřez z obrázku (viz. obr. 2.5). Ta se porovnává s původním obrázkem. Pokud v porovnávaném místě algoritmus určí jistou podobnost, která může být stanovena prahovou hodnotou, detekuje hranu objektu.

Použití tohoto způsobu kategorizace objektů se jeví velice výhodně díky vysoké úspěšnosti klasifikace na šedotónových obrazech. Navíc jsou tyto metody dostatečně odolné vůči šumu v obraze a jsou schopné dobře detekovat hrany na místě, kde se potkává více rozdílných textur. Nevýhodou je fakt, že pokud se bude maska aplikovat na obrázek v různých transformacích, případně budeme používat sadu více masek, tak se proces klasifikace velice zpomalí. Aby se časová náročnost výpočtu zmírnila, používá se v praxi např. maska s nižším rozlišením, nebo se opakovaně používá jen ta část masky, kde došlo ke změně.



Obrázek 2.5: Template matching. Převzato z [15]

2.4 HCI

Human-Computer Interaction (interakce člověk-počítač) [9] je vědní obor, zabývající se tvorbou uživatelských rozhraní. Počáteční rozvoj této vědní disciplíny se udál na přelomu 70. a 80. let 20. století jako reflexe na vývoj počítačů. Při tvorbě uživatelského rozhraní je nutné brát v potaz veliké množství faktorů, které jeho vývoj ovlivňují. Ať je to kupříkladu komunikace mezi pracovníkem v továrně a lisem nebo lékařem a rentgenem, jsou na každé z těchto uživatelských rozhraní kladeny různé požadavky. Proto v sobě tento obor musí seskupovat velké množství znalostí z různých specializovaných disciplín jako psychologie, lingvistika, ergonomie, informatika apod. Jsou definovány tři hlavní části:

- **Jedinec nebo skupina**
- **Počítač**
- **Způsob (jakým dva předešlé objekty spolupracují)**

Cílem je tedy vytvářet a vyvíjet prostředky komunikace, které jsou efektivní, jednoduché, intuitivní a také bezpečné.

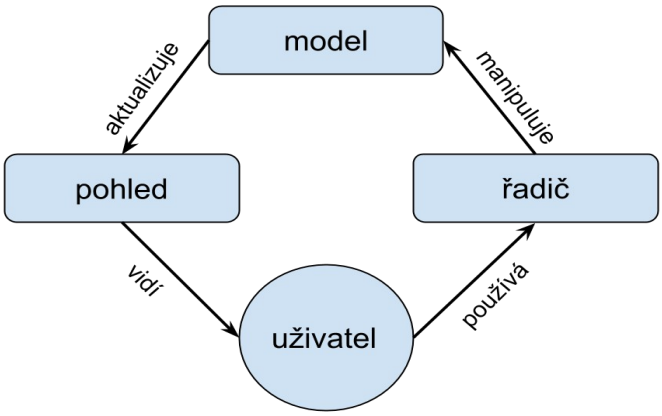
S rozvojem v této oblasti se dostává ke slovu pojem přirozené interakce (angl. Natural Interaction). Klasické ovládací prvky jako klávesnice a myš jsou pořád hojně využívané, avšak postupně zastarávají. Přirozená interakce člověka s počítačem zahrnující pouze uživatele samotného, jeho motoriku, pohyby a hlasové projevy, je pro uživatele pohodlnější a zefektivňuje vzájemnou komunikaci. V dnešní době je možno ji na dobré úrovni realizovat. Z velké části k tomuto kroku vpřed pomohl senzor Kinect, jehož hlavní výhodou je cena, která zapříčinila jeho velké rozšíření.

Model-View-Controller

Ve spojení s HCI je dobré uvést tzv. Model-View-Controller [8], což je softwarová architektura, která dělí roli funkčního komplexního celku do třech nezávislých komponent (viz. obr. 2.6):

- **Datový model (model)** – Obsahuje data a definuje chování aplikace (aplikační logiku), která souvisí s její doménou.
- **Pohled (view)** – Zobrazuje pohled na model v uživatelském rozhraní. Jinými slovy jsou data uchovaná v modelu prezentována uživateli.
- **Řadič (controller)** – Reaguje na vstup od uživatele a provádí změny pohledu nebo změny modelu.

Základním požadavkem tohoto vzoru je oddělení aplikační a prezentační logiky. Tímto krokem lze dosáhnout vetší spolehlivosti a možnosti použít prvek víceúčelově, a to hlavně při rychlém vývoji a velkém počtu změn. V případě prvku, který se stará o bezdotykové ovládání, je užití tohoto vzoru výhodné z hlediska možnosti přidávat nová gesta bez modifikace stávajícího řešení.



Obrázek 2.6: Model-view-controller

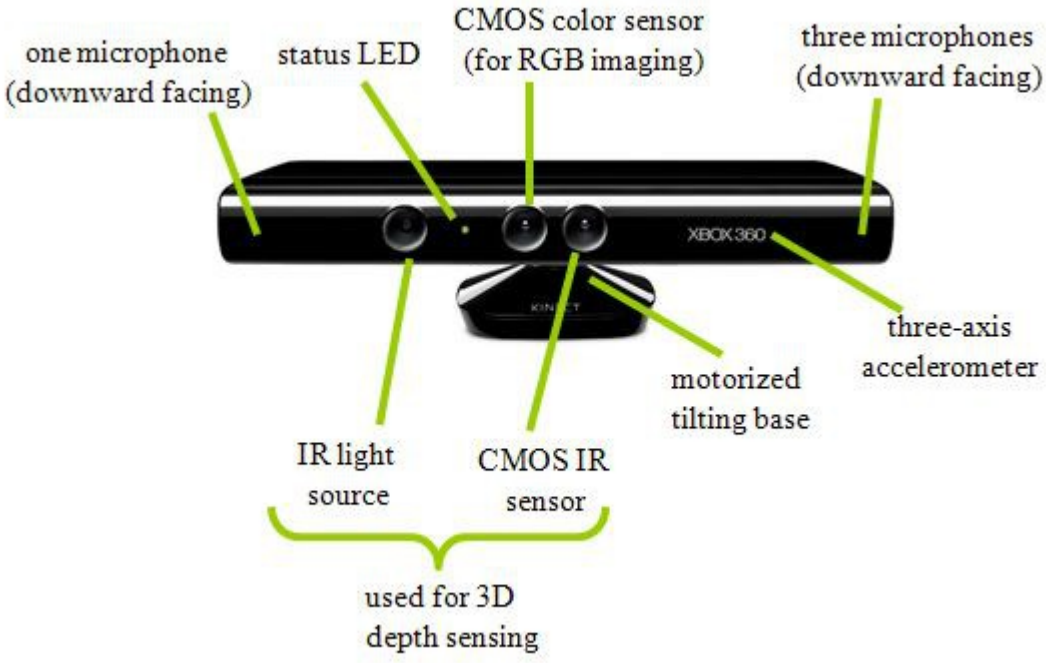
2.5 Kinect

Ke snímání pohybů uživatele je nutný snímač obrazu. Jedná se o technické zařízení, které měří danou fyzikální nebo technickou veličinu a tu převádí na signál, nejčastěji elektrický, jenž lze

dále přenášet a zpracovávat. Skládá se z objektivu tvořeného soustavou čoček zaostřující snímaný obraz a senzoru, zaznamenávající obrazovou informaci. Na trhu jsou běžně k dostání různé typy snímačů. Pro tento projekt bylo nutno se seznámit se zařízením Kinect (viz. Obr. 2.7) [10][11].

Kinect je původním zaměřením herní senzor, na jehož vývoji se podílela firma Microsoft ve spolupráci se společností PrimeSense, tehdy pojmenovávaným jako projekt Natal. Výsledný produkt se začal masově prodávat 4. listopadu 2010, již pod jménem „Kinect for Xbox 360“. Tehdy už bylo jasné, že vypuštění toto zařízení na trh je zlomovým okamžikem v oblasti uživatelských rozhraní.

Senzor je vybaven několika komponenty. Je to RGB kamera, hloubkový senzor, pole mikrofonů, servomotor v podstavci a čip PS1080.

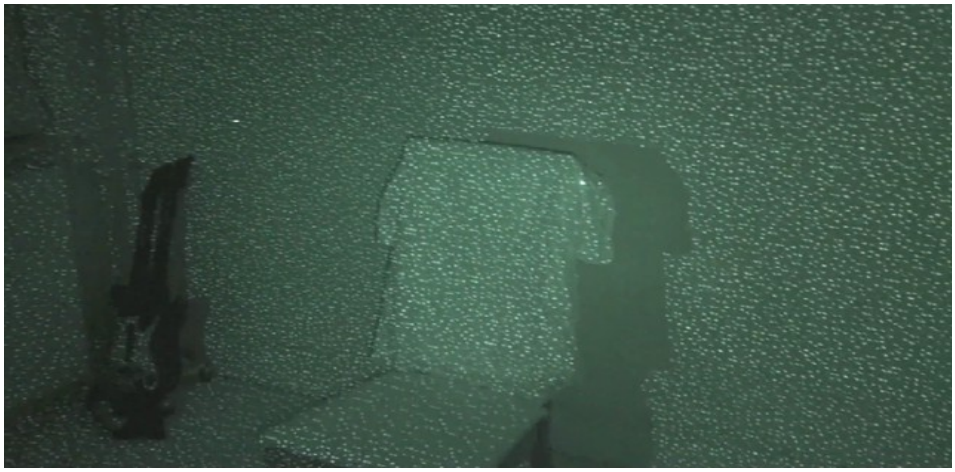


Obrázek 2.7: Zařízení Kinect. Převzato z [17]

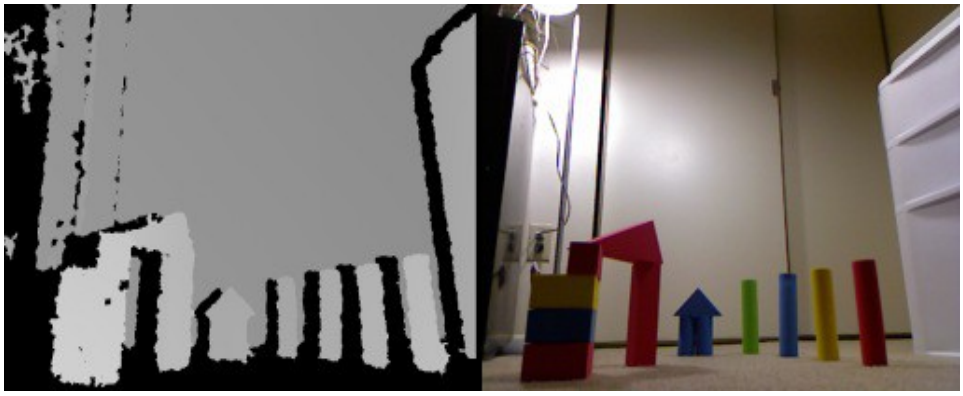
RGB kamera je standardní barevná CMOS kamera, která dokáže snímat barevný obraz v rozlišení 640x320 s frekvencí 30Hz a 24bitovou hloubkou (viz. obr. 2.9).

Hloubkový senzor, který je využit v této práci se skládá z dvojice IR projektoru a IR kamery. Projektor vysílá infračervené paprsky tvořící bodovou mřížku (viz. obr. 2.8). Infračervené záření se od objektů v cestě odrazí a je snímáno monochromatickou CMOS

kamerou. Minimální vzdálenost objektu od senzoru se pohybuje kolem hodnoty 0.8 metru. V této vzdálenosti dokáže senzor snímat plochu, která má rozměry přibližně 87 centimetrů na šířku a 63 centimetrů na výšku. Díky tomu je možno snímat například sedícího uživatele a reagovat na jeho pohyby, i když není zrovna v ideální pozici oproti senzoru. V minimální vzdálenosti od senzoru má Kinect také velmi dobrou rozlišovací schopnost 1,3 mm prostoru na jeden pixel. Na základě tohoto faktu lze s velkou přesností určit vzdálenost objektů ve scéně a pomocí vzdáleností mezi jednotlivými body vyslanými IR projektorem vytvořit hloubkovou mapu. Hloubková mapa má rozlišení 640x320 pixelů a 16bitovou hloubkou. Je vysílána frekvencí 30 Hz za sekundu (viz. obr. 2.9).



Obrázek 2.8: Paprsky z IR projektoru. Převzato z [16]



Obrázek 2.9: Vpravo obraz z RGB kamery, vlevo stejný obraz pomocí hloubkové mapy ve stupních šedi

Pole čtyř mikrofonů připevněných ve spodní části senzoru dokáže díky svému rozložení (jeden na levé straně tři na pravé) určovat směr, ze kterého zvuk přichází a tuto informaci využívat při rozpoznávání mluvících uživatelů.

V podstavci senzoru se vyskytuje servomotor, který umožňuje naklonit horní část Kinectu. Tímto mechanizmem se kompenzuje pozice (výška) umístění zařízení a zajistí se, aby pozorované objekty byly co nejvíce v zorném poli senzorů.

Čip PS1080 SoC od firmy PrimeSense je jádrem systému. Výpočetní logika v procesoru se stará o výpočet hloubkového obrazu scény ve VGA rozlišení a tyto data synchronizuje s daty z RGB kamery a zaznamenaným zvukem.

Komunikace s cílovým zařízením probíhá pomocí rozhraní USB 2.0. Avšak odběr zařízení je poněkud vyšší. Proto se pro připojení Kinectu s PC musí použít externí adaptér.

3 Návrh řešení

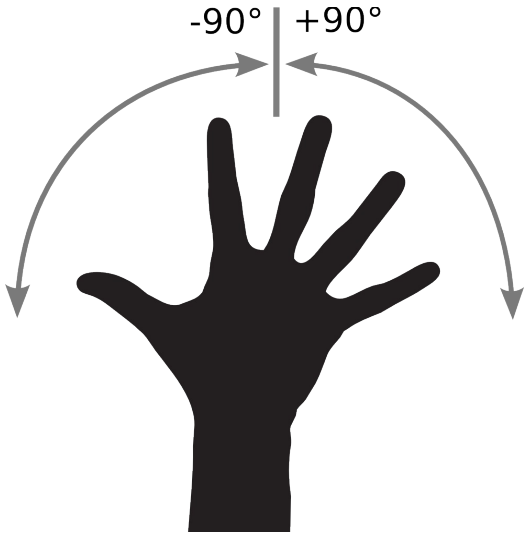
Tato práce je zaměřena na vytvoření aplikace, kterou je možno bezdotykově ovládat. Vyvíjený prvek má poskytovat uživatelské rozhraní umožňující zprostředkování komunikace mezi uživatelem a aplikací. Prvek by ve svém konečném výsledku měl usnadňovat ovládání aplikace tím, že nebude nutné pro některé úkony používat klasické ovládací média. Některé prvky ovládání aplikace budou realizovány gestem vytvořeným za pomocí rukou a prstů uživatele. K realizaci tohoto projektu bylo nutné zachytit obrazová data senzorem, ty pak následně zpracovavat a výsledek prezentovat v aplikaci.

Tato kapitola nabízí čtenáři informace o detailním rozboru práce, analýze požadavků zadání, problému, návrhu řešení dílčích funkčních částí celého prvku a vytvoření logické posloupnosti kroků, které vedou k dosažení požadovaného cíle. Následně je objasněna struktura a vysvětlena funkce jednotlivých částí podrobným popisem. Tento návrh je základem pro následnou implementaci.

3.1 Analýza zadaného problému

Stanovení cíle a prostudování zadání je podstatné a umožní vytvořit detailní návrh řešení. Cílem této práce je bezdotykové ovládání aplikace, respektive reakce ukázkové aplikace na pohyb ruky uživatele. Je tedy nutné určitým způsobem získat informace o pohybu uživatele, dále je zpracovat v počítači a výsledek předat v dané podobě dál ovládané aplikaci. Z toho plyne, že od zachycení gesta prováděného rukou senzorem, až po viditelné změny v reagující aplikaci, je provedeno veliké množství úkonů. K zachycení pohybu uživatele je nutný snímač, v tomto případě byl použit senzor Kinect for Xbox 360, který snímá uživatele. Tato data dále zpracuje centrální modul. Pokud uživatel provede předem stanovené gesto, tak tento modul z obrazových dat poskytnutých senzorem gesto rozpozná a zpracované informace popisující povahu gesta předá ukázkové aplikaci, která reaguje změnou hodnoty na ovládacím prvku.

Gesto, které je vytvořeno správným nasměrováním prstů uživatele, je softwarovou replikou reálné akce, kdy uživatel chce uchopit předmět ve tvaru knoflíku. Pro potřeby testování byla vytvořena aplikace s funkcí hudebního přehrávače, kde onen knoflík je prezentován jako ovladač (potenciometr) hlasitosti. Následná manipulace otáčením ruky ve směru a proti směru hodinových ručiček (viz. obr. 3.1) ve stavu uchopení potenciometru má za následek snižování nebo zvyšování hlasitosti přehrávané hudby.



Obrázek 3.1: Gesto ovládání aplikace

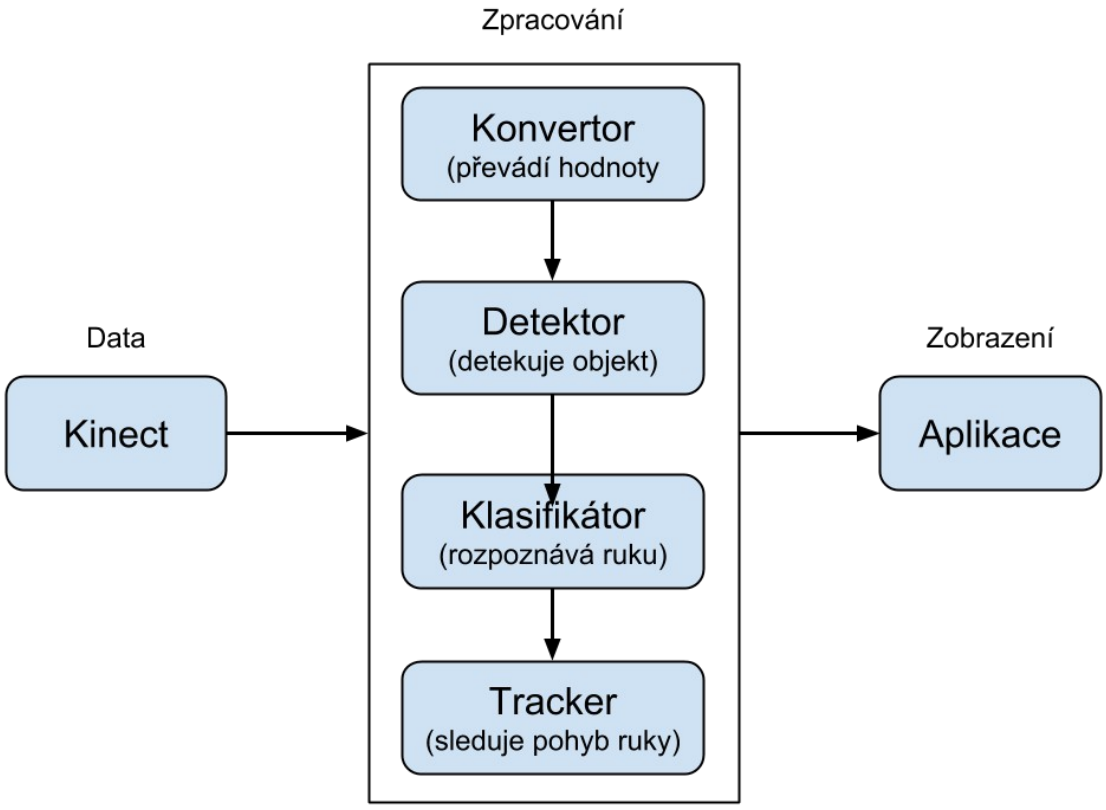
Tento prvek může náležitým způsobem pomoci všem lidem v zjednodušení ovládání aplikace. Přece jenom ovládání přirozeným způsobem je v jistých ohledech pohodlnější a intuitivnější, než používání klávesnice či myši. Využití by si aplikace mohla najít také u lidí s postižením, které nemohou z nějakého důvodu používat klasické ovládací prvky. Komplexnější řešení by poté mohlo nalézt uplatnění na veřejných místech, jako jsou letiště, pošty či nemocnice, kde využití bezdotykového ovládání by vedlo k zlepšení hygienických podmínek.

3.2 Koncept řešení

Následující část definuje strukturu návrhu řešení daného problému. Budou diskutovány jednotlivé bloky, jejich vnitřní skladba a vliv na celkovou funkcionalitu (viz. obr. 3.2).

Kompletní řešení se dá shrnout do tří vzájemně propojených bloků. První částí je hardwarové zařízení, v tomto případě kamera, která snímá uživatele a zprostředkovává tak informaci o jeho pohybu, kterou digitalizuje a posílá ke zpracování. Druhou částí je jádro celého projektu, které se stará o zpracování obrazových dat. Aplikuje funkce pro práci s obrazem, až ze vstupního datového toku získá požadovanou informaci o pohybu uživatele. Informace je poslána poslední třetí části, jíž je koncová aplikace. Ta v sobě implementuje uživatelské rozhraní v grafické podobě, a proto převede obdržená data do vizuální formy pro koncového uživatele.

Část, ve které probíhá zpracování, se skládá z několika dílčích modulů. Konvertor slouží k převodu hodnot poskytovaných senzorem do jiné reprezentace na hodnoty, které se budou lépe zpracovávat dalšími moduly. Detektor dokáže z obrazových dat vybírat objekty nejblíže senzoru. Maska objektu a jeho obálka jsou výstupem této části. Klasifikátor funguje na principu porovnávání. Normalizované obrázky masek porovnává s bankou obrázků. Tento proces rozhoduje, zda maska obsahuje objekt ruky, či nikoliv. Nakonec modul trackeru provádí sledování detekované ruky a zjišťuje změnu úhlu natočení objektu mezi jednotlivými snímky. Obsahuje také optimalizaci, která limituje extrémní (nepravděpodobné) hodnoty a zpřesňuje tak výsledky publikované trackerem.



Obrázek 3.2: Diagram bloků

3.3 Snímání uživatele

Na vstupní části návrhu je soubor obrazových dat. Jejich zdrojem je senzor Kinect. Ten snímá pohyb a natočení rukou uživatele. Vzhledem k faktu, že Kinect nedisponuje pouze klasickou RGB kamerou a integruje v sobě i hloubkový senzor, který má dobrou přesnost, byla

využita v tomto řešení sekvence hloubkových obrazových dat, které Kinect poskytuje v předepsaném formátu. Parametry jednotlivých bodů obrazu, v tomto případě jejich souřadnice ve 3D prostoru a vzdálenost od senzoru, jsou dostačující potřebnou informací. Avšak nesmíme zapomenout, že přesnost zařízení hloubkového senzoru Kinect je doprovázena různými vadami v přenášeném obraze. Některé techniky segmentace obrazu mohou poskytovat značně zkreslené výsledky, pokud jim jsou poskytnuta nepřesná a zkreslená vstupní data. Je tedy dobré provést minimalizaci těchto vad v obraze.

Snímací zařízení posílá další části sekvenci snímků rychlostí přibližně 30 snímků za sekundu. V kombinaci s rozlišením 640 na 480 obrazových bodů a 32 bitové hloubce je to solidní objem dat, který se musí zpracovat a je nutné dbát na to, aby navržené algoritmy nebyly pouze přesné, ale taky časové efektivní. Na druhou stranu je díky plynulosti a kvalitě videosekvence poskytované Kinectem možné jednodušeji a přesněji pracovat s obrazem a získávat z něj informaci.

Použití pouze hloubkového datového toku z Kinectu má navíc také výhodu v tom, že odpadne nutnost kalibrovat zařízení. Obyčejně, pokud se používá informace z RGB kamery a zároveň i z hloubkového senzoru, je nutná kalibrace, aby každá z kamer poskytovala totožný obraz. Ovladače pro Kinect sice základní kalibrační nastavení obsahují, ale pokud požadujeme od aplikace maximální přesnost, tak se kalibraci nevyhneme.

3.4 Moduly zpracování obrazu

Následující podkapitola popisuje jádro celého systému. Část, která se skládá ze čtyř modulů implementuje techniky předzpracování, segmentace a vyššího zpracování.

Konvertor

S nasnímanými hloubkovými daty je rozumné pracovat jednotným způsobem. Data poskytovaná hloubkovým senzorem se dají zpracovat do matice, která má rozměry odpovídající rozlišení senzoru. Zařízení Kinect poskytuje několik datových proudů, ve kterých jsou informace o obraze reprezentována různými veličinami s různým datovým typem. Datové informace každého pixelu (většinou obsahuje vzdálenost od senzoru) je možné vhodným způsobem překonvertovat, například na souhrn hodnot od nuly až do čísla 255. Pokud matici v tomto okamžiku zobrazíme, dostaneme osmibitový obrázek ve stupních šedi (viz. obr. 3.3), kde místa blízko senzoru jsou světlejší a místa dále od senzoru tmavší. Výhoda převodu hodnot

spočívá také v možnosti zobrazit výsledek po postupném aplikování technik pracujících s obrazem a lépe vybrat efektivnější metodu.

Obrázek 3.3: Hloubkový obrázek z kinectu převedený do šedotónového osmibitového obrázku

Detektor

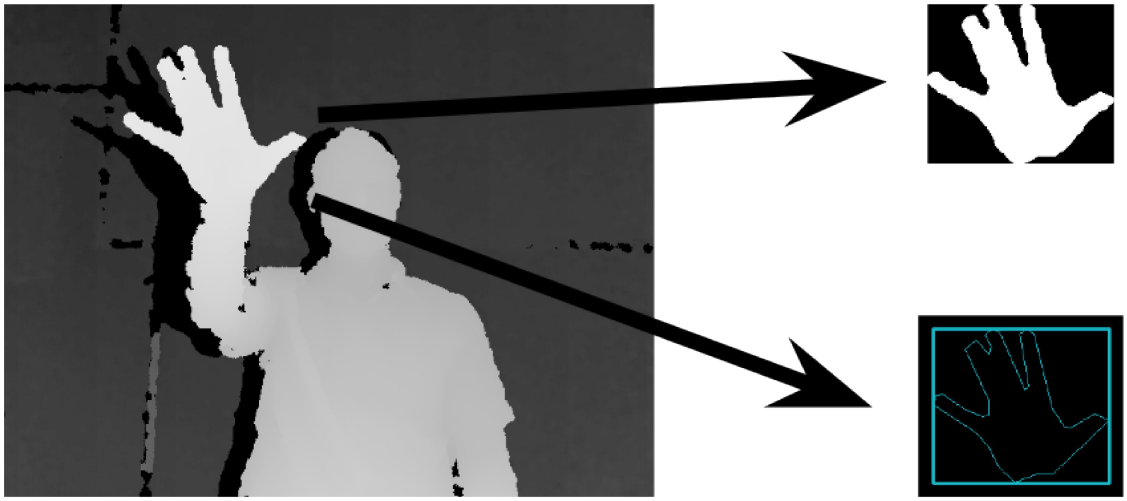
Získání pozice dlaně a prstů uživatele je klíčové. Pokud chce uživatel provést gesto, je časté, že pozice jeho ruky, resp. dlaně a konečků prstů je oproti zbytku těla nejblíže k senzoru. Zjištění nejnižší hodnoty pixelu v obrazové matici při aplikaci gesta uživatele poskytne pixel v obraze, který se pravděpodobně nachází na některém z prstů. Takto získaný bod lze použít jako startovací uzel algoritmu Flood fill, který, jak jeho název napovídá, funguje na principu rozšiřování oblastí. V případě vztažené ruky uživatele v pozici k provedení gesta mají body znázorňující ruku v obrazové matici velice podobnou hloubku, tím pádem i stupeň šedi. Tohoto poznatku lze využít v nastavení cílové barvy pro algoritmus, která se od barvy startovního bodu bude lišit minimálně, a v ideálním případě získat tzv. masku obsahující ruku uživatele.

Vzhledem k faktu, že ruka je součástí objektu celého člověka je možné, že algoritmus by po obarvení segmentu ruky pokračoval dále a přebarvil i celý zbytek těla. Tomuto stavu se dá zabránit vhodně zvoleným typem a hodnotou prahu, která bude závislá na startovacím pixelu. Takto je možno vzdálenější části objektu z obrazu odfiltrovat a získat tak obraz pouze s dlaní a prsty.

Maska objektu, která je výstupem algoritmu Flood fill, je černobílá matice, kde jedna z barev reprezentuje pozadí a druhá objekt (viz. obr. 3.4). Z tohoto obrázku, kde je jasně zřetelná

hrana mezi objektem a pozadím, lze získat tzv. bounding box objektu (viz. obr. 3.4). V případě tohoto typu obrazu udává bounding box souřadnice a rozměry rámečku, který co nejtěsněji obepíná a ohraničuje objekt v masce.

Jako vstup pro metodu aktivních kontur je možné použít zmíněný bounding box. Tato obálka je algoritmem s určitou přesností přimknuta k objektu. Na výstupu algoritmu dostaneme tedy uzavřenou křivku, která co nejtěsněji kopíruje hranu objektu.



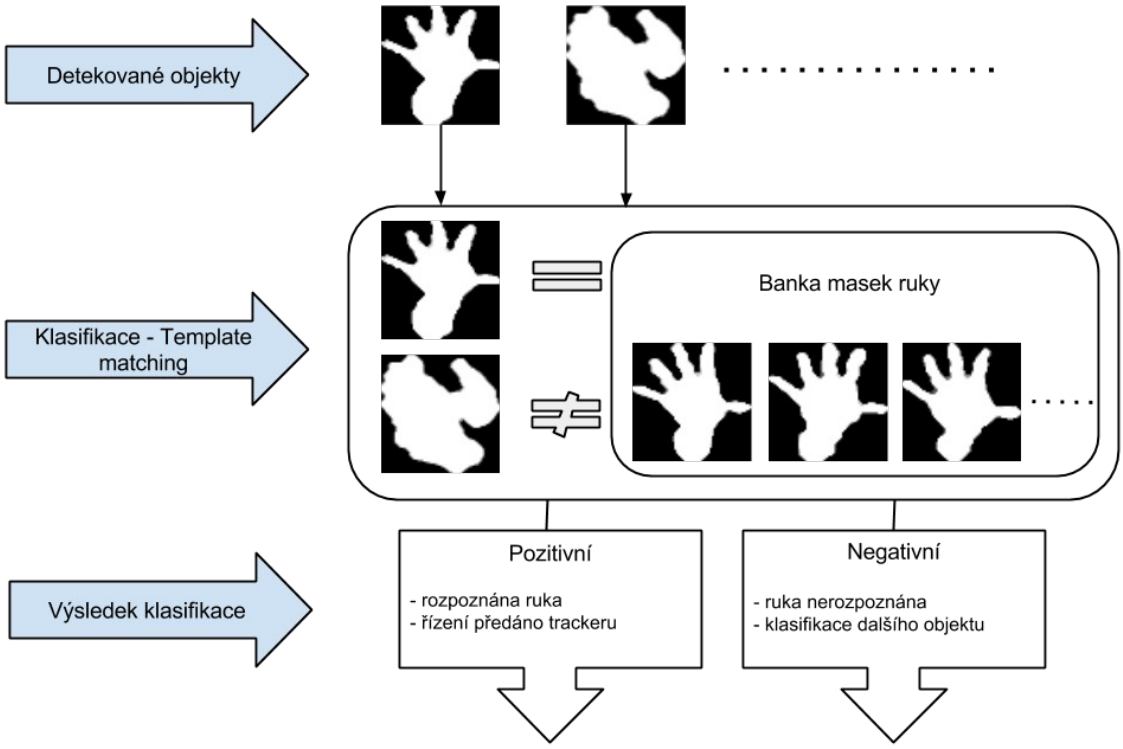
Obrázek 3.4: Maska a obálka ruky uživatele

Klasifikátor

Předpokladem je, že na zkoumaném objektu respektive jeho části leží bod, který je nejblíže snímacímu zařízení. Pokud je ovšem uživatel ve stavu, kdy neprovádí žádné gesto a ruka není nejblíže snímači, výstup algoritmu Flood fill poskytne masku, která obsahuje objekt, na němž leží v tu chvíli nejbližší bod ke snímači. Pokud by například nastala situace, že snímač zaměří hlavu uživatele a ten s ní bude pohybovat, mohl by program vyhodnotit toto chování jako pohyb ruky a tím pádem měnit hlasitost v testovací aplikaci. Proto jsou tyto objekty nežádoucí a je třeba je odfiltrovat. Je nutné zjistit, kdy maska obsahuje ruku uživatele a kdy nikoliv.

Klasifikace objektu se v tomto případě skládá ze dvou tříd. V jedné třídě jsou objekty, které se podobají ruce, v druhé zbylé objekty detekované algoritmem. Rozřazení do těchto tříd je realizováno porovnáním (viz. obr. 3.5). Jelikož k porovnání použijeme sadu obrázků zobrazující pouze ruku a maska obsahuje také pouze ruku, ovšem je rozměrově větší, tak se z masky se provede výřez místa, kde je objekt. Informaci o místě, kde se má vyříznutí aplikovat s sebou nese bounding box. Výřez z masky a banka několika vzorků, které by zobrazovaly ruku v pozici pro provedení gesta, se normalizují. Pak lze provést porovnání. To zda by vzorek patřil

do oné třídy reprezentující objekty ruky lze zhodnotit například sumou počtu shodných pixelů mezi výřezem masky a vzorky a vhodně nastavenou mezní hodnotou mezi jednotlivými třídami.



Obrázek 3.5: Klasifikace objektů

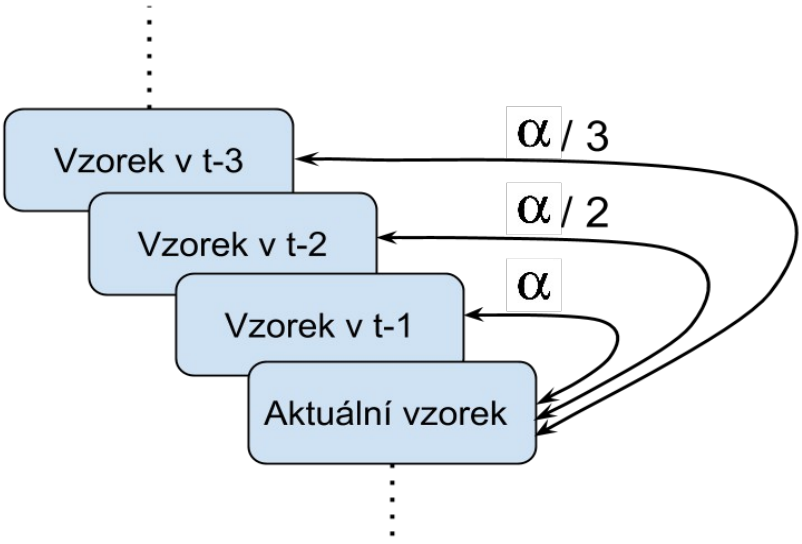
Tracker

Na situaci, kdy objekt v masce je klasifikovaný jako ruka navazuje další jeho sledování.

Poté, co uživatel vztáhne ruku k senzoru a ten ji detekuje, nejčastěji provede gesto. Zařízení Kinect je schopné poskytovat hloubková data s rychlostí třicet snímků za sekundu. Tato frekvence je k získání přesné trajektorie pohybujícího se objektu dostatečná.

Pruběh gesta, které je definováno k ovládání aplikace, v čase, je možno popsat jako skupinu masek ruky, které jsou vůči sobě úhlově pootočeny. V tomto případě, kdy nás zajímá změna natočení ruky uživatele, postačí zjistit změnu úhlu mezi počátečním a koncovým stavem ruky, respektive posun mezi jednotlivými maskami. V praxi to lze provést tak, že jeden z obrázků je statický a druhý vůči němu rotujeme. Po každém pootočení zjistíme porovnáním počet nesouhlasných pixelů. Natočení, které bude mít nejmenší počet těchto pixelů, vybereme a tak zjistíme úhel.

Výsledek získaný tímto způsobem jde kumulovat. Technika popsaná výše se provede dvakrát, ovšem po druhé se obrázky mezi sebou vymění. Tím dojde k zpřesnění. Další optimalizace spočívá v porovnání vzorků, které mezi sebou těsně nesousedí (viz. obr. 3.6). Při každém nově příchozím obrázku proběhne porovnání onoho obrázku kromě předchozího i s dalšími dvěma předešlými. Získané úhly se zprůměrují a zahrnou se do výsledku. Tato optimalizace prostým průměrováním minimalizuje výskyt vysokých hodnot změny úhlu, které mohou vznikat při chybném chodu trackeru.



Obrázek 3.6: Optimalizace trackeru, kde alfa je úhel změny natočení a t je čas aktuálního vzorku.

4 Realizace a testování

Nasledující kapitola popisuje softwarové a hardwarové prostředky použité při vytváření aplikace. Dále obsahuje popis konfigurace nástrojů. Vysvětluje, jakým způsobem byl naimplementován popisovaný návrh. Pojednává také o metodice testů, jejich průběhu a vyhodnocení.

4.1 Použité nástroje a jejich konfigurace

Výsledná aplikace je navržena jako multiplatformní. Pro vývoj bylo použito na straně hardware zařízení kinect Xbox 360 v kombinaci s PC, na straně software OS Linux Ubuntu verze 12.04 LTS. Jako softwarové prostředky pod uvedeným OS byly zvoleny ROS verze Hydro, knihovna OpenCV verze 2.4.8 a editor Eclipse 4.3.1. Koncová aplikace byla navržena ve frameworku Qt 5.2.

Softwarové prostředky

Pokud se zaměříme na zařízení Kinect (viz. Kap. 2.5), zjistíme, že je k dispozici několik ovladačů pro práci s tímto zařízením. Kupříkladu firma Microsoft vyvinula Software developement kit (SDK) pro Kinect. S tímto kitem pracuje veliké množství vývojářů, a proto je možné na internetu, či jiných veřejných zdrojích najít spousty různě kvalitních aplikací. Ty se dají velice dobře použít pro studium a pochopení podstaty fungovaní výsledného systému.

Tento projekt je ovšem vytvořen za použití nástrojů, které nejsou pro tento účel tak hojně používány. Tím ovšem není míněno, že by tyto nástroje byly méně kvalitní. Pro přiblížení následující text popíše softwarové prvky použité pro realizaci.

ROS

Robot Operating System neboli ROS[1], je softwarový produkt poskytující knihovny a nástroje, které pomáhají vývojářům při vytváření aplikací. Tento systém je velice globální a obsahuje nepřeberné množství prvků. Vývojář pracující v ROS má tedy vše, na co si vzpomene, ovšem daní za to je z počátku velice obtížná orientace v systému. ROS má velikou výhodu v tom, že obsahuje ovladače pro Kinect, a proto se data z Kinectu dají získat velice přívětivou cestou v

1 Viz http://www.ros.org/

předepsaném formátu. Data v prostředí ROS se dají ukládat do tzv. Rosbag a později zpětně přehrávat, což se ocení zejména v oblasti testování vyvíjené aplikace. Jádrem celého systému jsou takzvané vlákna či témata (topics) a uzly (nodes). Uzly mezi sebou komunikují pomocí zpráv (messages), které jsou zasílané přes vlákna. Vlákno může obsahovat i více než jeden uzel, který data do vlákna zapisuje (subscriber) a jeden, co z něho čte (publisher). Díky této hierarchii je možné mít velice dobrou kontrolu na celým projektem, hlavně komunikací – přenášenými daty, mezi jednotlivými moduly výsledného řešení.

OpenCV

S daty z Kinectu se dá pracovat mnoha způsoby. Pro zpracování obrazových dat je vhodné použít multiplatformní knihovnu pro manipulaci s obrazem zvanou OpenCV[2] (Open source Computer Vision). Tato knihovna obsahuje nepřeberné množství funkcí pro práci s obrazem, jejichž vlastní implementace by byla neefektivní a zbytečná. Výhoda této knihovny tkví v tom, že je pod open-source licencí BSD a dá se tady zdarma použít například pro studijní účely. Primárně je navržená pro práci v prostředí s jazyky C a C++, není ale problém ji využít s generátorem také v jazycích Python či Octave. Hloubkový nebo RGB obraz z kinectu obsahuje parazitní jevy jako například šum. OpenCV poskytuje celou řadu funkcí, které dokáží tyto jevy brát v potaz a provádět s obrazem požadované operace.

Důležité je zmínit tzv. CvBridge, což je modul, který dokáže převádět obrazová data publikovaná Kinectem ve formě streamu do interpretace obrazu, jakou používá knihovna OpenCV pomocí třídy cv::Mat. Zajišťuje tak bezproblémovou práci s obrazovými daty, které jsou uchovány ve formě matice.

Konfigurace nástrojů

V následující části je popsáno, jak nastavit základní softwarové prostředky, aby bylo možné návrh implementovat a zprovoznit.

Připojení zařízení Kinect do PC

Zařízení komunikuje s PC přes klasické usb 2.0. Pro jeho napájení však nestačí a je vyžadován externí zdroj. Zda je Kinect připojen, se dá v Linuxu zjistit příkazem *lsusb,* který vypíše informace o USB portech a zařízeních, která jsou na ně připojena.

2 Viz http://opencv.org/

Instalace ROS

ROS byl do OS Linux Ubuntu nainstalován a nastaven pomocí následujících příkazů:

- ***sudo apt-get install ros-hydro-desktop-full*** - *N*ainstaluje systém ROS obsahující všechny potřebné balíčky a utility k vývoji.
- ***sudo rosdep init* a *rosdep update*** - Nastaví nástroj rosdep, který je nutný pro běh některých komponent jádra systému
- ***source /opt/ros/hydro/setup.bash*** - Tento příkaz je nutné spouštět při každém novém spuštění shellu nebo ho přidat do .bashrc, abychom měli přístup k příkazům ROS.

V této fázi je již možné systém ROS spustit příkazem *roscore* a poté ho plně využívat.

ROS a Kinect

ROS poskytuje balíček *openni_launch*. Ten obsahuje ovladače OpenNI pro Kinect a umožňuje konvertovat hloubková data zachycená Kinectem na hloubkové obrazy, a to i bez kalibrace. Spuštění ovladače je možné přes příkaz *roslaunch openni_launch openni.launch.* V této fázi ROS začne komunikovat s Kinectem a poskytne uzly a vlákna, s nimiž je možné pracovat. Jejich seznam se získá příkazem *rosnode list* pro uzly nebo *rostopic list* pro vlákna.

ROS a OpenCV

ROS komunikuje s knihovnou OpenCV pomocí CVBridge. Tato utlilita dokáže převádět streamy dat vytvářené Kinectem, a umožňuje tak bozproblémové čtení dat a ukládání do matice.

Catkin

Každý nový projekt v ROS je dobré vytvořit ve formě balíčku. K tomu slouží nástroj Catkin. Příkaz *catkin_create_pkg <název_balíčku>* vytvoří základní strukturu balíčku, se kterou se dále jednoduše pracuje. Díky balíčku získáme možnost pracovat s uzly a vlákny, lepší kontrolu nad projektem (jeho závislosti) a možnost používat další nástroje ROS.

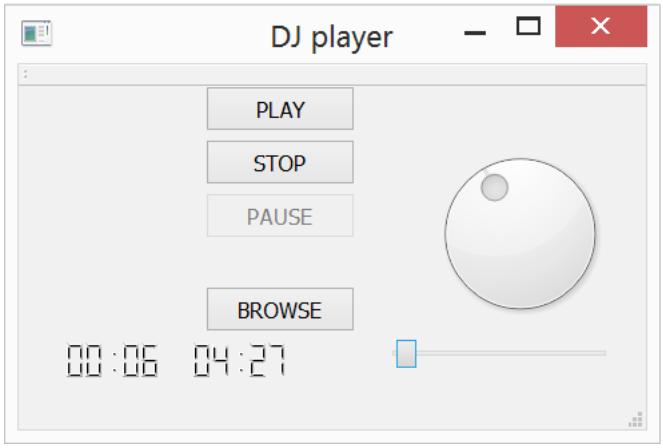
Rosbag

Je balíček, který v sobě obsahuje nástroje pro záznam a přehrávání dat na zvoleném vláknu. Tato funkcionalita se dá dobře využít pro nabírání a uschování testovacích dat. Příkazem *rosbag*

record <název_vlákna> kinectdata začne nahrávání dat z vybraného vlákna do souboru kinectdata.

4.2 Ukázková aplikace

Pro otestování správné funkčnosti implementovaného řešení, byla vytvořena testovací aplikace (viz. obr. 4.1). Tato aplikace s názvem DJ player je svým ovládacím rozhraním, funkcemi a chováním velice blízká klasickému hudebnímu přehrávači s tím rozdílem, že je ochuzená od všech pokročilých funkcí a umožňuje pouze základní operace pro práci s hudebním souborem. Zjednodušení je zde hlavně z toho důvodu, aby byla aplikace co nejkompaktnější, jednoduchá na ovládání a dobře sloužila k hlavnímu účelu – ukázce funkčnosti řešení.



Obrázek 4.1: Testovací aplikace

4.3 Popis implementace programu

Konvertor (převod hodnot)

Tato část se skládá ze dvou hlavních bloků. První blok obsahuje užití cv_bridge. V programu odchytáváme hloubkový stream *camera/depth/image*, který poskytuje Kinect. Pomocí funkce *cv_bridge::toCvCopy* převedeme obrázek z interpretace ROS do OpenCV. Ve chvíli, kdy máme obrazová data ve formátu matice pro OpenCV, druhý blok provede převod jednotlivých hodnot pixelů matice z metrů reprezentovaných Float32 do Int8. Takto vznikne obraz ve stupních šedi s 256 různými úrovněmi, kde nejbližší pixel k senzoru má nejvyšší hodnotu.

Detektor (detekce objektu)

Nejprve najdeme nejbližší bod k senzoru. To se provede standardním průchodem matice a nalezení bodu s nejvyšší hodnotou. Použijeme funkci m*inMaxLoc()*, která vrátí pozici pixelu s nejvyšší hodnotou v matici. Tento pixel se využije jako počáteční bod pro funkci *floodFill()*. Tato funkce začne rozšiřovat oblast od startovního bodu až po nastavený práh. Jako výstup této funkce je maska, což je binární matice, kde číslo jedna reprezentuje oblast zasaženou algoritmem Flood fill. Maska, která je velikostí stejná jako původní obrázek, se ořízne. Nejprve v masce najdeme pomocí algoritmu *findContours()* konturu objektu v masce. Z ní jsme schopni zjistit nejtěsnější obdélníkovou obálku objektu. Tuto obálku najdeme za použití funkce *boundingRect().*

Klasifikátor (rozpoznání ruky)

Oříznutá maska objektu se klasifikuje. K rozpoznání objektu ruky se porovnávaný objekt normalizuje pomocí *resize()*. Klasifikátor využívá k porovnání 25 normalizovaných vzorků levé i pravé ruky. Provede se porovnání za pomocí funkce *compare()*. Výsledkem je matice, kde jsou nesouhlasné body po konvoluci provedenou funkcí *compare()* nastaveny na hodnotu jedna. Funkce *countNonZero()* zjistí počet těchto pixelů. Po provedení porovnání výřezu masky se všemi vzorky banky se vybere nejnižší hodnota počtu pixelů získaných funkcí *countNonZero()*. V programu nastavená prahová hodnota rozhodující o tom, zda počet nesouhlasných pixelů řadí vzorek do jedné ze skupin, byla zjištěna praktickým testováním programu.

Tracker (změna natočení)

Pokud je klasifikace pozitivní (je nalezen vzorek ruky), tak následuje tracking. Aktuální a předešlý vzorek ruky se porovnávají, aby se zjistil úhel natočení. Tracking využívá podobné funkce jako klasifikátor. Aktuální vzorek postupně otáčí. V rozsahu +-20 stupňů aplikuje po dvou stupních funkci r*otate()*, která přepočítá jednotlivé pixely ve vzorku a otočí ho. Po každém natočení jsou aplikovány funkce *compare()* a *countNonZero()* jako u klasifikátoru. Nejnižší hodnota získaná *countNonZero()* určí, jak velký bude směr a úhel natočení.

Optimalizace sledování ruky

Pro zpřesnění výpočtu a zmenšení chyby, která může vznikat při sledování otáčení ruky byly implementovány dvě optimalizace.

Jedna spočívá v kumulování výsledků. Jak je popsáno v návrhu, celý proces natáčení obrázku se počítá pro každé natočení dvakrát, kdy jeden ze vzorků je statický a druhý se natáčí a obráceně. Implementačně se tato optimalizace realizuje jednoduše, pouze prohozením parametrů, kterými jsou porovnávané vzorky.

Druhou optimalizací je provádění porovnání mezi více vzorky. V podstatě se základní porovnání aplikuje i na několik předešlých vzorků, které jsou uložené v bufferu (*std::vector*). Získané úhly se průměrují. Získáme tak stabilnější hodnotu.

4.4 Testování

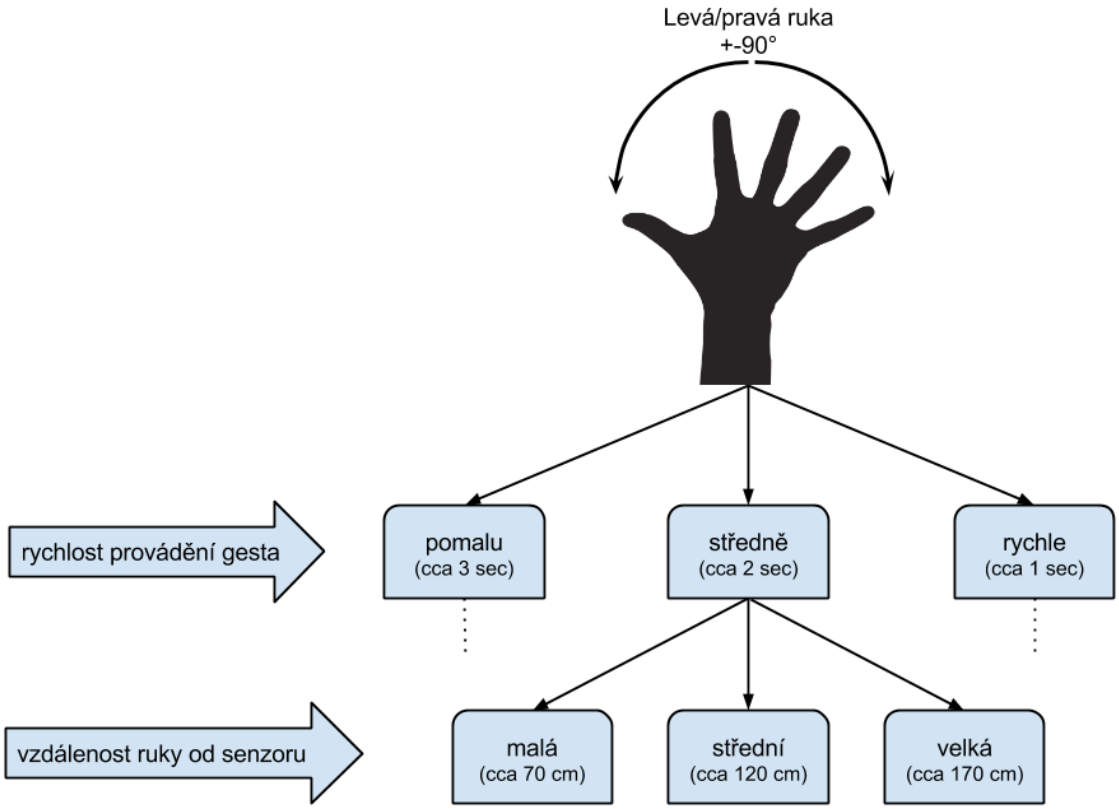
Vyvíjený prvek byl podroben testům, aby se zjistilo, jak funguje v praxi. Vytvořila se testovací metodika.

Testovací sada

Základní otázkou bylo, co chceme testovat a jak to testovat. Z průzkumu, jak nejlépe ověřit navržený program vyplynulo, že budeme sledovat dva jevy. To, zda probíhá správně detekce rukou a jak přesný je tracker.

Byl vytvořen soubor gest, kterými se bude testovat. Čtyři základní gesta natočení ruky se budou provádět různou rychlostí otáčení, v různé vzdálenosti od senzoru. Celkem tak vznikne 36 vzorků (viz obr.4.3).

S rozhodnutím, že testy budou více komplexnější a znovupoužitelné vznikla nutnost data nahrát, aby mohla být později použita k automatickému testování. Zde je právě výhoda ROS a balíčku rosbag, který umí jednoduše nahrávat data z Kinectu. K nahrávání byl vybrán jak hloubkový obraz poskytovaný vláknem *camera/depth/image*, tak obraz z RGB kamery na vlákně *camera/rgb/image_color*.



Obrázek 4.2: Testovací sada

Sběr testovacích dat

Testovací data byla nahrávána v laboratoři. Vytvořily se podobné podmínky, při jakých může běžný uživatel prvek používat. Pět lidí nezávisle testovalo prvek tím, že nejdříve dostali pokyny k provádění gest, které poté realizovali. Vznikl tak soubor 180 vzorků získaných pro automatické testy.

Testovací skript

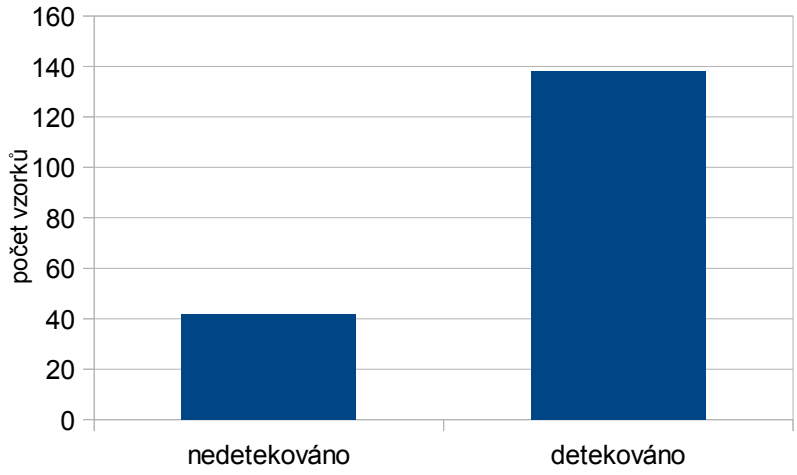
Testování probíhalo automatizovaně pomocí skriptu. Aby se využilo výhod ROS, byl vytvořen balíček s názvem tester pomocí nástroje catkin. Pro realizaci se použil jazyk C. Program postupně spouští jednotlivé bagy. Kromě toho také komunikuje s hlavním programem pomocí signalizačního vlákna, kde informuje program o událostech jako je začátek či konec bagu.

Výsledky testů

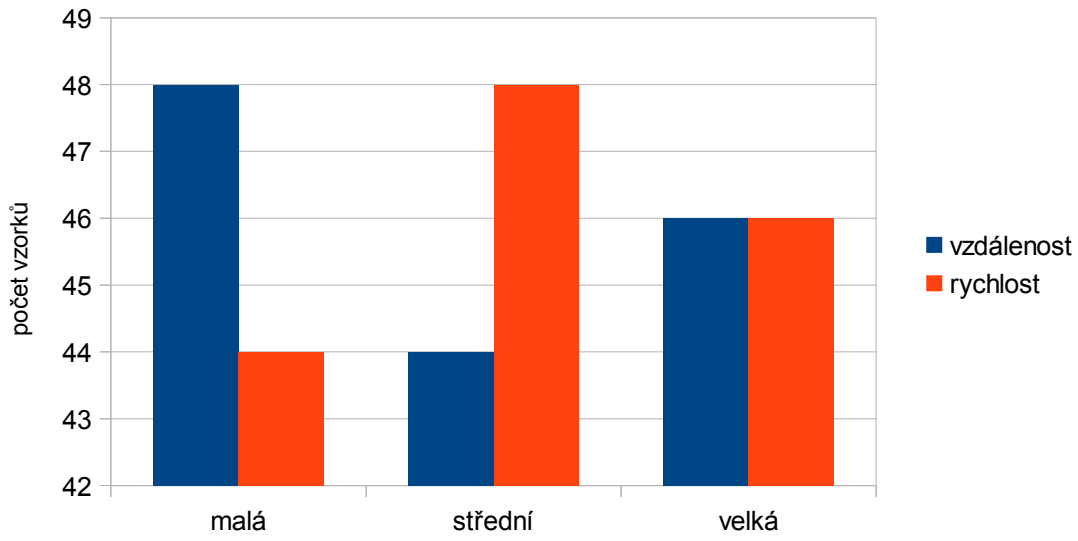
Pro názornou ukázku výsledku testování byla výstupní data zpracována graficky.

Výsledky úspěšnosti detekce ruky (viz. obr. 4.3) jsou 42 nedetekovaných oproti 138 správně detekovaných vzorků. Což je přibližně 77% úspěšnost. Analýza výsledků detekcí ruky byla provedena také s ohledem na povahu dat a to z pohledu úspěšnosti detekce v různé vzdálenosti ruky od senzoru a při různé rychlosti provádění gesta (viz. obr. 4.4). V tomto případě je z grafu patrné, že detekce ruky probíhá na celém pásmu testovaných dat s velice podobnou úspěšností. Po vizuální kontrole několika bagů, a s přihlédnutím na výsledky automatického testování, má největší podíl na neúspěšnosti detektor objektů, který předpokládá, že nejbližší bod senzoru je ruka, což v praxi nemusí vždy platit.

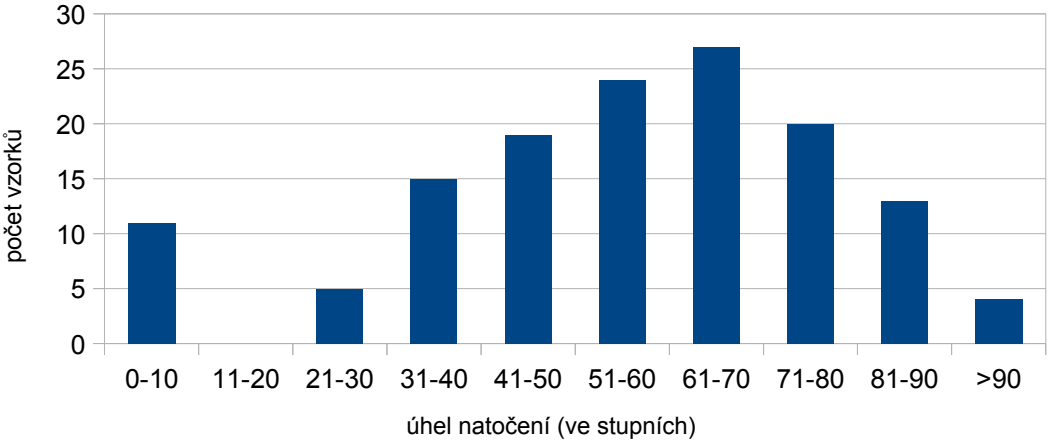
Přesnost trackeru ukazuje graf na obrázku 4.5. V grafu jsou již potlačeny nulové hodnoty, kdy neproběhla správně detekce ruky. Průměrná hodnota úhlu natočení ruky zachycená trackerem na testovacích datech bez chybných detekcí je přibližně 60,5 stupně, což je odchylka 29,5 stupně od výchozí hodnoty 90 stupňů. V těchto výsledcích se však projevuje lidský faktor, protože člověk nedokáže odhadnout přesně 90 stupňů a natočení je tím pádem nepřesné. I přes tuto chybu dosahuje přesnost hodnoty přibližně 67%. Tato přesnost je negativně ovlivněna ztracením sledované ruky v počátku pohybu (levý sloupec grafu přesnosti viz obr. 4.5) a nespolehlivostí při rozpoznávání konce gesta (pohybu ruky), kdy uživatel již gesto dokončil, ale tracker ruku ještě sleduje a vznikají tak parazitní hodnoty, které se promítnou do výsledku.



Obrázek 4.3: Graf úspěšnosti detekce



Obrázek 4.4: Graf detekce v závislosti na vzdálenosti a rychlosti



Obrázek 4.5: Graf stability

5 Závěr

Tato práce si kladla za cíl vytvořit prvek, který bude implementovat základní funkci uživatelského rozhraní, ovládání potenciometru hlasitosti pomocí gesta vytvářeného rukou uživatele.

Řešení zadání práce zahrnovalo získat obecné informace o problematice a poté nastudovat metody pro zpracování a segmentaci obrazu. Seznámení s vývojovými prostředky a pochopení jejich funkčnosti bylo také klíčové.

Studium metod, vytvoření návrhu a implementace je detailně zdokumentováno v předešlých kapitolách této technické zprávy. Od souhrnu teoretických znalostí v podobě rozboru použitých metod přes návrh ve formě modulů až po implementační část včetně testování finálního řešení .

Možnosti další úpravy a vylepšení se přímo nabízí. V rámci zajímavého rozšíření by mohl program detekovat obě ruce. Druhá ruka by se dala použít jako specifikace, co se bude zesilovat. Gesta modelované touto rukou by mohla představovat vícepásmový ekvalizér. Podle toho kolik by uživatel vztyčil prstů by pak probíhalo zvýraznění vždy jiného frekvenčního pásma.

Literatura

[1] Morris, Tim.: *Computer Vision and Image Processing*, Palgrave Macmillan, 2004, ISBN 978-0-333-99451-1.

[2] Hlaváč V., Šonka M.: *Počítačové vidění*, Grada, Praha, 1992, ISBN 80-85424-67-3.

[3] Hlaváč V., Šonka M., Boyle R.: *Image processing, Analysis, and Machine Vision*, PWS Publishing, Pacific Grove, 1999, ISBN 0-534-95393-X.

[4] Russ J. C.: *The Image Processing Handbook*, CRC Press, 2002.

[5] Wikipedia: *Flood fill*. [online]. 2009 [cit. 2014-05-04], dostupné z: <http://en.wikipedia.org/wiki/Flood_fill>.

[6] Španěl M., Beran V.: *Obrazové segmantační techniky.* [online]. 2006 [cit. 2014-05-04], dostupné z WWW <http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769325>.

[7] Kotek Z., Mařík V., Hlaváč V., Psutka J., Zdráhal Z.: Metody rozpoznávání a jejich aplikace, Academia, Praha 1993, ISBN 80-200-0297-9.

[8] Baray C.: *The model-view-controller (MVC) design pattern.* [Online]. 1999 [cit. 2014-05-06], dostupné z: <http://cristobal.baray.com/indiana/projects/mvc.html>.

[9] Carrol J. M.: *HCI models, theories, and frameworks: toward a multidisciplinary science*, Morgan Kaufmann, San Francisco 2003, ISBN 1-55860-808-7.

[10] Wikipedia: *Kinect.* [online]. 2014 [cit. 2014-05-07], dostupné z: <http://en.wikipedia.org/wiki/Kinect>.

[11] Microsoft: *Kinect Sensor Specifications*. [online]. 2012 [cit. 2014-05-04], dostupné z: <http://msdn.microsoft.com/en-us/library/hh855355.aspx>.

[12] Bílý V.: Detekce objektu pomocí význačných bodů [Online]. 2012 [cit. 2014-05-02]. Obrázek ve formátu PNG. Dostupné z: <http://medusa.fit.vutbr.cz/wiki/index.php/Detekce_objektu_pomoci_vyznacnych_bodu>

[13] Saad M.: Solving Problems with Recursion. [Online]. 2004 [cit. 2014-05-06]. Obrázek ve formátu JPG. Dostupné z: <http://www.devshed.com/c/a/practices/solving-problems-with-recursion/>.

[14] OpenCV.: Basic thresholding operations. [Online]. 2011 [cit. 2014-05-12]. Obrázek ve formátu JPG. Dostupné z: <http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html>

[15] OpenCV.: Template matching. [Online]. 2011 [cit. 2014-05-06]. Obrázek ve formátu JPG. Dostupné z: <http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html>

[16] Ayucar I.: Some technology behind Kinect [Online]. 2010 [cit. 2014-05-15]. Obrázek ve formátu PNG. Dostupné z: <http://graphicdna.blogspot.cz/2010/11/some-technology-behind-kinect.html>

[17] Gregori E.: Building a Kinect Based Robot for under $500.00 [Online]. 2011 [cit. 2014-05-14]. Obrázek ve formátu PNG. Dostupné z: <http://buildsmartrobots.ning.com/profiles/blogs/building-a-kinect-based-robot-for-under-500-00>

Příloha A

Obsah CD

- /text – technická zpráva ve formátu ODT a PDF
- /src – zdrojové kódy aplikace
- /bags – výběr z testovacích dat
- /video – demonstrační videosekvence
- /poster – demonstrační plakát
- /README – manuál k překladu a spuštění aplikace